VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2015 Dominik Ther

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Dominik Ther**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: ABB s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Kot, Ph.D.**

Konzultant bakalářské práce: Bc. Michael Filsák

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 21.4.2015 .....

Rád bych na tomto místě poděkoval mé rodině za podporu při studiu. Dále firmě ABB s.r.o. a jejím zaměstnancům za možnost vykonat odbornou praxi.

V neposlední řadě také Vysoké škole báňské a mému vedoucímu panu Ing. Martinu Kotovi, Ph.D.

Abstrakt

Tato bakalářská práce popisuje mou odbornou praxi ve firmě ABB s.r.o. V průběhu praxe jsem vystupoval v pozici vývojáře, který měl za úkol implementovat informační systém pro testování znalostí zaměstnanců. Jako hlavní technologie jsem použil framework ASP.NET MVC, PetaPoco a nástroj RoundhousE. Dále je v této práci popsána metodika pro efektivní vývoj a nástroj pro její podporu.

Klíčová slova: Individuální odborná praxe, ABB, Informační systém, C#, ASP.NET MVC, PetaPoco, RoundhousE

Abstract

This bachelor thesis describes my professional practice in the ABB s.r.o. company. As a software developer my task was to implement information system for knowledge testing of employees. As the main technology I used ASP.NET MVC framework, PetaPoco and RoundhousE tool. Furthermore, this bachelor thesis describes the methodics for effective development and tool to support it.

Keywords: Individual professional practice, ABB, Information system, C#, ASP.NET MVC, PetaPoco, RoundhousE

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
ASP.NET	–	Framework pro vytváření webů
CSRF	–	Cross-site request forgery
CSS	–	Cascading Style Sheets
DDL	–	Data Definition Language
DML	–	Data Manipulation Language
GTP	–	Generating test project
HTML	–	HyperText Markup Language
IIS	–	Internet Information Services
JSON	–	JavaScript Object Notation
MIT	–	Massachusetts Institute of Technology
MSSQL	–	Microsoft SQL Server
MSP	–	Motor service portal
MVC	–	Model - View - Controller
např.	–	například
ORM	–	Object Relational Mapping
PDF	–	Portable Document Format
SQL	–	Structured Query Language
TFS	–	Team Foundation Server
tzv.	–	takzvaný
T-SQL	–	Transact - Structured Query Language
XML	–	Extensible Markup Language
XSS	–	Cross-site scripting

Obsah

1 Úvod **4**

2 Popis odborného zaměření firmy a popis pracovního zařazení studenta **5**
2.1 Firma ABB s.r.o. 5
2.2 Popis pracovního zařazení studenta ve firmě ABB s.r.o. 5

3 Projekt GTP - Generating test project **6**
3.1 ASP.NET MVC 6
3.2 Návrh uživatelského rozhraní 6
3.3 TFS 8
3.4 SCRUM 8
3.5 RoundhousE a databáze 9
3.6 ORM - PetaPoco 9
3.7 Webový klient 10
3.8 Optimalizace 16
3.9 Dokumentace 17
3.10 Nasazení 17

4 Uplatněné a chybějící znalosti **19**
4.1 Uplatněné znalosti a dovednosti získané v průběhu studia 19
4.2 Chybějící znalosti a dovednosti v průběhu odborné praxe 19

5 Závěr **20**

6 Reference **21**

Seznam obrázků

1 Struktura projektu a zvýraznění složek MVC 7
2 Část návrhu uživatelského rozhraní v programu Microsoft Visio 2013 . . . 7
3 Část PDF dokumentu s testem. 16
4 Část View s vyhodnoceným testem. 17

Seznam výpisů zdrojového kódu

1 Soubor App.config 10
2 Ukázka syntaxe Razor 10
3 Výčtový typ pro typy otázek 11
4 Použití DataAnnotation 11
5 Metoda GetAll() ve třídě CategoryModel 12
6 Metoda GetAll() ve třídě BaseModel bez komentářů 12
7 Metoda Insert() ve třídě QuestionController 14
8 Část metody pro generování výsledků v PDF 15

1 Úvod

Bakalářskou praxi jsem si vybral nejen z důvodu získání nových znalostí a rozšíření těch stávajících, ale také kvůli získání praxe v oboru, která je dnes u absolventů téměř nutná. Zajímalo mě také jak fungují procesy uvnitř firmy a s čím se tedy mohu potencionálně setkat při zaměstnání. Kontaktoval jsem vybrané firmy, které tuto praxi poskytovaly. Z těchto firem mě nejvíce zaujala firma ABB s.r.o., u které jsem se rozhodl vykonat svou bakalářskou praxi.

S firmou jsem si tedy domluvil první pohovor, který vedl Ing. Ján Mináč. Zajímal se především o mé předchozí zkušenosti a znalosti ze školy. Byly mi sděleny základní informace o vývoji a použitých technologiích v softwarovém oddělení. Jednalo se například o jazyk C# a práce s MSSQL. To byl jeden z důvodů pro zvolení této firmy. Po absolvování pohovoru jsem dostal sadu testů z různých oblastí, které měly dát firmě hrubý náhled mých aktuálních znalostí. U těchto testů mi také pomohly znalosti získané ve škole a z osobního zájmu o obor. Ten samý den mi byl sdělen výsledek a nabídnuta možnost vykonat u této firmy bakalářskou praxi. Vzhledem k tomu, že jsem byl první student, který při pohovoru dostal sadu testů a firma tento postup chtěla opakovat, bylo mi dáno za úkol tento proces převést do informačního systému, který by zjednodušil další práci.

V prvních kapitolách vás seznámím s firmou ABB s.r.o. a mým zařazením. Následně budu popisovat používané technologie, vývoj a postup samotné praxe. V posledních kapitolách této práce shrnu mé působení ve firmě, získané, nebo chybějící znalosti a celkový závěr.

2 Popis odborného zaměření firmy a popis pracovního zařazení studenta

2.1 Firma ABB s.r.o.

ABB je mezinárodní firma působící v oblasti energetiky a automatizace. Pracuje pro ni více než 140 000 lidí ve více než 100 zemích světa. ABB také působí v České republice a to od roku 1970, kde pro ni pracuje více než 3 400 lidí [1]. V současné době působí v osmi lokalitách. Firma se skládá z pěti divizí: Výrobky pro energetiku, Systémy pro energetiku, Automatizace výroby a pohony, Výrobky nízkého napětí a Procesní automatizace. Většina produktů z České republiky míří na export. To svědčí o kvalitě výrobků a produktů, které se zde vyrobí. Více si o firmě ABB můžete přečíst na její oficiální webové stránce [2].

2.2 Popis pracovního zařazení studenta ve firmě ABB s.r.o.

Praxi jsem vykonával v ABB s.r.o. Ostrava. Konkrétně v softwarovém oddělení, které patří pod divizi procesní automatizace. Jak bylo zmíněno v úvodu, firma potřebovala systém na správu testů a jejich vyhodnocení s možností budoucího rozšíření. Systém měl umožňovat správu otázek a odpovědí, kdy otázky a odpovědi mohly být různého typu. Například kombinace otázky s odpovědí, která měla na výběr z více možností, přičemž typy se mohly mezi sebou kombinovat. Dále měl systém umožnit generování testů pro budoucí zaměstance nebo studenty, generování jejich výsledků, kontrolu testů, přehled výsledků a jejich porovnání. Další drobné požadavky přicházely v průběhu praxe. Dostal jsem tedy samostatný projekt. Vedení tohoto projektu dostal na starost tým, který vyvíjel aplikaci MSP. Vývojáři z tohoto týmu pro mě v rámci praxe vystupovali v roli zákazníka a také jako konzultanti při řešení úkolů. Na začátku bylo třeba specifikovat prostředí, ve kterém se bude aplikace vyvíjet, a jaké technologie budou použity. Po diskuzi s týmem mi byly pro vývoj systému jako hlavní technologie doporučeny ASP.NET MVC4 a SQL Server od společnosti Microsoft. Další použité technologie budou popsány v následujících kapitolách.

3 Projekt GTP - Generating test project

První den praxe jsem byl obeznámen s denním chodem firmy a představen týmům ze softwarového oddělení. Následně mi bylo přiřazeno místo pro výkon praxe a byl mi zadán první úkol. Zadaným úkolem bylo nastudování technologie ASP.NET MVC [3]. Bylo tedy nutné si přečíst a vyzkoušet tutoriály na oficiálních stránkách. Seznámil jsem se tedy s ASP.NET MVC, syntaxí Razor a s praktickou implementací vzoru MVC, který je implementován také do dalších projektů ve firmě.

3.1 ASP.NET MVC

ASP.NET je zdarma dostupný framework pro vytváření dynamických webových stránek za pomoci HTML, CSS a jazyka JavaScript. Pro můj systém byl zvolen model ASP.NET MVC zahrnující použití vzoru MVC, který se skládá ze tří rozdílných komponent (Model, View, Controller).

Model - udržuje data aplikace a je schopen se starat o tok dat mezi aplikací a databází.

View - slouží k zobrazení dat a interakci s uživatelem.

Controller - Controller propojuje dvě výše uvedené komponenty. Dokáže zpracovat data, která buď přichází od uživatele, nebo jsou vrácena modelem. Veškerá získaná data je pak Controller schopen odeslat komponentě View. Je také schopen například vracet výsledek ve formátu JSON.

Vzor MVC také zjednodušuje vývoj. Tím, že jsou tyto tři části od sebe odděleny, mohou na nich vývojáři pracovat nezávisle. Výhodou je integrace s Visual Studiem se kterým jsem se setkal již v době mého studia a v rámci praxe jsem ho využil nejen jako vývojové prostředí, ale také pro správu verzí kódu a úkolů na projektu.

Na obrázku č. 1 můžete vidět, že jsou jednotlivé komponenty modelu MVC rozděleny do složek (Models, Views, Controllers), zároveň je zde vidět struktura projektu. Projekt se celkově skládá ze čtyř dalších projektů.

DataAccess - obsahuje ORM framework PetaPoco pro přístup k databázi, který bude popsán v kapitole 3.6.

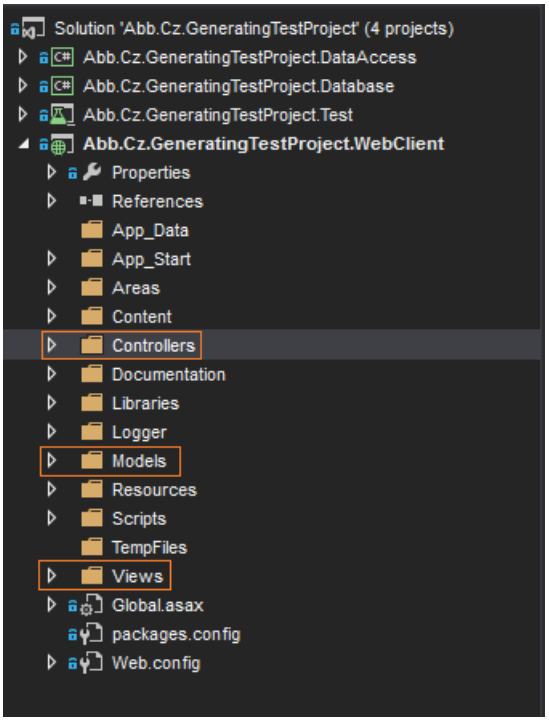
Databse - slouží ke správě SQL skriptů pomocí nástroje RoundhousE, který bude popsán v kapitole 3.5.

Test - projekt do kterého se budou psát případné testy webového klienta.

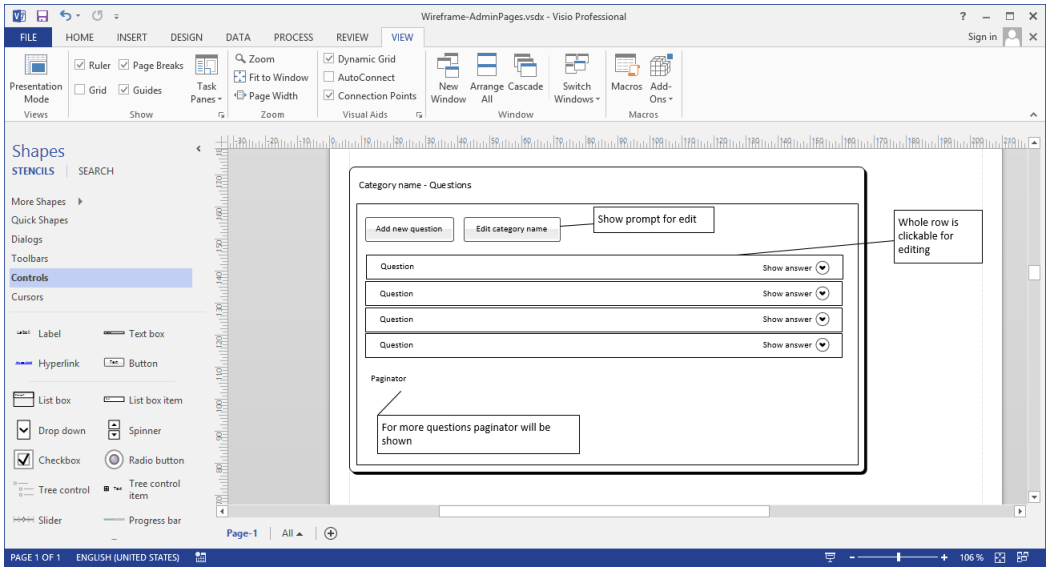
WebClient - webový klient v ASP.NET MVC.

3.2 Návrh uživatelského rozhraní

Po nastudování ASP.NET MVC mi byl zadán další úkol. Navrhnout uživatelské rozhraní, tzv. wireframe. Vytvořil jsem tedy návrhy stránek pro administrátorskou i uživatelskou část systému. Tyto návrhy byly dále diskutovány s týmem do doby, než jsem úkol dopracoval do finální podoby. Na obrázku č. 2 můžete vidět část návrhu uživatelského rozhraní v programu Microsoft Visio 2013 (dále jen Visio), který se ve firmě používá.



Obrázek 1: Struktura projektu a zvýraznění složek MVC



Obrázek 2: Část návrhu uživatelského rozhraní v programu Microsoft Visio 2013

3.3 TFS

TFS je produkt společnosti Microsoft pro verzování zdrojového kódu a poskytuje nástroje pro podporu agilních metodik. V mém případě to byla agilní metodika Scrum.

Verzování systému má tu výhodu, že máme k dispozici různé verze projektu a je pro nás jednodušší provádět případné změny. Můžeme z těchto verzí dělat odvozené projekty nebo balíčky. Každý programátor na projektu má na svém počítači svou lokální verzi projektu. Po implementování určité funkcionality nahraje svou verzi do TFS, kde je k dispozici pro ostatní programátory ke stažení. Nahrání na TFS se nazýva *check-in*.

Před nahráním je možno nastavit **Code review**. Code review je kontrola kódu ostatními členy týmu. Je vhodné zejména pro kontrolu chyb v kódu, abychom na TFS zbytečně nenahrávali chybné verze. Pokud v kódu najdeme chybu, je možné ji okomentovat a poslat programátorovi zpět na opravení. Zároveň se tímto můžeme naučit programovací techniky například od zkušenějších členů týmu.

Nastavil jsem tedy TFS projekt pro můj systém a propojil ho s Visual Studiem. Code review jsem posílal týmu, který mě měl na starost. Ze začátku mi byl často kód posílán zpět s připomínkami a musel jsem provádět refaktorizaci. Refaktorizace je proces, při kterém přepisujeme kód do přehlednější podoby s důrazem na ošetření výjimek. Postupem času jsem si osvojil některé techniky a kód psal přehledněji.

3.4 SCRUM

Při projektu GTP jsem byl také postupně seznamován s metodikou Scrum. Scrum je agilní softwarová metodika, jak vyvíjet software, a slouží ke zlepšení procesu dodávání softwaru. Tato metodika je založena na iteračních cyklech, kterým se říká **Sprint**. Na začátku vývoje se sepíší požadavky od zákazníka do takzvaného **Product Backlog** a setřídí podle priority. Toto má na starosti **Product Owner**, který zároveň zodpovídá za celý projekt. Každý Sprint je definován na určitou dobu a obsahuje vybrané položky z Product Backlogu. Položky se vybírají před začátkem Sprintu v plánování, kterému se říká **Sprint Planning**. Dále jsou zde takzvané denní schůzky **Daily scrum**, které slouží k naplánování dne a shrnutí dne předešlého. Na konci Sprintu se naplánuje retrospektiva. To má za úkol **Scrum master**, který se stará o vývojový tým a měl by mu pomáhat řešit problémy, které by tým mohly zdržovat při vývoji. Retrospektiva slouží týmu pro zlepšení procesů a zhodnocení předešlého Sprintu. Výsledkem by měl být seznam věcí, které chceme zlepšit a které chceme zachovat. Po retrospektivě se znovu naplánuje Sprint Planning, dokud nebudou všechny položky z Product Backlogu hotovy.

Pro podporu této metodiky jsem využil TFS, prováděl zde potřebné změny a vyzkoušel si tuto metodiku na svém projektu, jelikož se mnou ke konci spolupracovali i jiní programátoři. Provedl jsem nastavení Product Backlogu, vytváření úkolů a plánování. Při plánování jsme využívali *scrum karty*, které nám pomáhaly odhadnout složitost daného problému. Tato práce v týmu pro mě byla cennou zkušeností. V průběhu praxe mi byla umožněna účast na školení, které bylo zaměřeno na správné plánování a odhadování složitosti dílčích úkolů ve Scrumu.

3.5 RoundhousE a databáze

3.5.1 RoundhousE

Dalším úkolem byl návrh databáze a vytvoření potřebných SQL skriptů. V této fázi jsem se musel seznámit s nástrojem RoundhousE [4], který slouží k verzování databáze a automatickému nasazení. Použití toho nástroje je velmi jednoduché a výhodné. Pro RoundhousE jsem vytvořil samostatný projekt, který obsahuje hiearchii složek pro SQL skripty a spustitelné soubory pro vytvoření nebo aktualizaci databáze. Nejčastěji používané složky jsou *up, functions, sprocs*. Dále jsou to spustitelné soubory *DropCreateDatabase.bat* a *UpgradeDatabase.bat*.

up - složka s DDL/DML skripty, které jsou provedeny pouze jednou a to při vytvoření databáze. Pokud se v nějakém souboru, který již byl pomocí RoundhousE spuštěn, provedly změny, nasazení databáze se neprovede a zobrazí se chybová zpráva. Proto je vždy nutné vytvořit nový soubor pro každou změnu databáze. Z toho důvodu se tyto skripty číslují, např. *0001_Create_Table.sql*, *0002_AlterTable.sql*, aby bylo zachováno pořadí. Skript *0001_Create_Table.sql* se provede jako první.

function - složka se skripty s funkcemi. Pokud jsou závislé na pořadí, seřadíme je abecedně.

sprocs - složka se skripty s uloženými procedurami

DropCreateDatabase.bat - pokud existuje specifikovaná databáze, provede její smazání (DROP) a vytvoření (CREATE). Zároveň provede všechny SQL skripty v daném pořadí podle priorit složek a názvů souborů.

UpgradeDatabase.bat - spustí nově přidané skripty ve složkách, které se spouštějí pouze při vytvoření databáze. V ostatních složkách spustí všechny skripty.

Nástroj RoundhousE si pamatuje verze databáze. Podle data a času jsme schopni vyhledat jaké změny byly provedeny, případně se vrátit k předchozí verzi. Tento nástroj má samozřejmě mnoho nastavení. (Pokud se chcete dozvědět více, navštivte oficiální stránku RoundhousE, kde je odkaz na podrobnou dokumentaci.)

3.5.2 Databáze

Návrh databáze jsem provedl v programu Visio, kde je možné znázornit tabulky, sloupce, vazby a určitá integritní omezení. Po diskuzi s týmem, kdy jsme došli k prvotnímu návrhu, jsem přešel ke konfiguraci nástroje RoundhousE a napsal SQL skripty pro vytvoření tabulek.

3.6 ORM - PetaPoco

Po vytvoření databáze a tabulek bylo třeba pro přehlednost vytvořit nový projekt, ve kterém je použito PetaPoco [6]. PetaPoco je jednoduchý a rychlý ORM framework s jedinou třídou. Automaticky generuje ORM, poskytuje pomocné metody a obsahuje SQL Builder pro jednoduché sestavení SQL dotazů. Dále poskytuje sadu atributů, které umožňují jednodušší práci s modely. Byl vyvinut skupinou vývojářů Topten Software a je poskytován

zdarma pod licencí Apache License, Version 2.0 [5]. (Více se o samotném frameworku, jeho využití a nastavení můžete dozvědět na webových stránkách Topten Software.)

PetaPoco jsem nainstaloval přes Package Manager Console ve Visual Studiu příkazem *Install-Package PetaPoco*. Dalším krokem bylo vytvoření souboru *App.config*, který můžete vidět ve výpisu kódu č. 1. Tento soubor obsahuje informace o přípojení k databázi, tzv. *connection string*. Po nainstalování a nastavení se musí znovu uložit soubor *Database.tt*, který je součastí frameworku PetaPoco. Po jeho uložení se vygeneruje ORM podle databáze, která je uvedena v *connection stringu*. Pokud již ORM máme vygenerováno, ale změnili jsme databázi, například přidání sloupce do tabulky, stačí znovu uložit soubor *Database.tt*. V tuto chvíli bylo vytvořeno vše potřebné, abych mohl začít pracovat na webovém klientovi.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="ConnectionString" providerName="System.Data.SqlClient" connectionString="
        Data Source=(localdb)\SQLEXPRESS;Initial Catalog=GTPProjectLocalhost;Integrated
        Security=SSPI;MultipleActiveResultSets=True;" />
  </connectionStrings>
</configuration>
```

Výpis 1: Soubor App.config

3.7 Webový klient

Dalším úkolem bylo vytvoření webového klienta. Pomocí Visual Studia jsem tedy vytvořil nový ASP.NET MVC projekt se syntaxí Razor. Razor je značkovací jazyk, který se používá v HTML a umožňuje provádět kód na stránce. Nejčastěji pomocí tohoto jazyka vypisuji data na stránce. Ve výpisu kódu č. 2 je uvedena část HTML kódu s jazykem Razor, který začíná znakem @ a na stránce vypíše obsah property *Name* předaného modelu.

```
<thead>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
    </tr>
</thead>
```

Výpis 2: Ukázka syntaxe Razor

3.7.1 Language resources

V celém projektu používám takzvané Resources Files. Jedná se o XML soubor s příponou *.resx*. Ukládají se do něho záznamy v podobě klíč - hodnota a poznámka. Tyto soubory používám k překladu, jelikož jeden z požadavků byl, aby aplikace byla jednoduše použitelná i ve více jazycích. Klíč slouží jako jedinečný identifikátor záznamu a hodnota je samotný obsah, který má být zobrazen. Ukládaní v těchto souborech má velkou výhodu pro budoucnost aplikace. Nemusíme hledat textové řetězce v celém projektu, ale máme je

na jednom místě. Resource Files jsou také známé mezi překladateli a existuje pro ně mnoho nástrojů. Není tedy problém nechat si tyto soubory přeložit například od specializované firmy.

3.7.2 Modely

Dalším úkolem bylo vytvoření modelů. Jeden model odpovídal jedné databázové tabulce. Obsahoval metody CRUD (Create, Read, Update, Delete), jiné pomocné metody a tzv. convert metody. Convert metody sloužily na převod vlasního modelu na objekt, který přijímá PetaPoco. V modelech se také mohly vyskytovat výčtové typy tzv. *enum*. Ty bylo vhodné použít například pro různé typy otázek pro ulehčení práce s kódem. Ve výpisu kódu č. 3 můžete vidět příklad takového výčtového typu.

```
public enum QuestionTypeEnum
{
        OpenQuestion,
        CodeQuestion,
        TwoCodeQuestion
}
```

Výpis 3: Výčtový typ pro typy otázek

Každý model měl property, které odpovídaly sloupcům dané tabulky a navíc mohl obsahovat jiné pomocné property nebo proměnné. Nad jednotlivé property, kde to bylo vyžadováno, bylo potřeba nadepsat DataAnnotation. DataAnnotations mohou v ASP.NET sloužit například k validaci nebo usnadní práci s vykreslením View. Ve výpisu kódu č. 4 můžete vidět příklad property s více DataAnnotation. Lze je tedy kombinovat. První DataAnnotation s klíčový slovem *Display* nám ulehčí práci s View, kdy poznáme jaký název pro naší property se má použít. Druhá DataAnnotation s klíčovým slovem *Required* určuje povinost při vyplňování. A třetí s klíčovým slovem *StringLength* určuje maximální délku řetězce. Můžete si také všimnout použití Resource Files.

```
[Display(Name = "CategoryName", ResourceType = typeof(LanguageResources))]
[Required(ErrorMessageResourceName="Required", ErrorMessageResourceType=typeof(
    LanguageResources))]
[StringLength(100, ErrorMessageResourceName="LenghtTo100", ErrorMessageResourceType=
    typeof(LanguageResources))]
public string  Name { get; set; }
```

Výpis 4: Použití DataAnnotation

Ve výpisu kódu č. 5 můžete vidět metodu *GetAll()* v třídě CategoryModel. Nejen tato metoda byla impementována téměř ve všech modelech a tak bylo vhodné vytvořit tzv. BaseModel, který obsahoval často opakující se metody a pomocné metody. Jedna z pomocných metod byla *GetTableName<TData>()*, která vrací pouze název tabulky. Nelíbilo se mi řešení, kdy názvy tabulek byly posílány jako řetězec a pomocí metod PetaPoca jsem navrhl jednodušší řešení, kdy nám stačí pouze PetaPoco model. Metody byly napsány pomocí generických typů tak, aby je bylo možno volat z kteréhokoliv modelu. Ve výpisu kódu č. 6 můžete vidět implementaci metody *GetAll<TModel, TData>()* v BaseModelu

a také volání metody *GetTableName<TData>()*, se kterou jsem v metodě mohl vynechat jeden parametr a vyhnul se případným chybám v názvu tabulek.

```
/// <summary>
/// Get all categories from database
/// </summary>
/// <returns>List of all categories</returns>
public static List<CategoryModel> GetAll()
{
    return BaseModel.GetAll<CategoryModel, Category>(ConvertDataToModel);
}
```

Výpis 5: Metoda GetAll() ve třídě CategoryModel

```
public static List<TModel> GetAll<TModel, TData>(Func<TData, TModel>
     ConvertDataToModel, int id = 0, string queryWhere = "", string queryOrder = "")
    where TModel : class
    where TData: class
{
    ConnectionStringDB m_db = new ConnectionStringDB();
    List<TModel> models = new List<TModel>();

    Sql sql = Sql.Builder
        .Select("*")
        .From(GetTableName<TData>());

    if (! string.IsNullOrWhiteSpace(queryWhere) && id != 0)
    {
        sql.Where(queryWhere, id);
    }

    if (! string.IsNullOrWhiteSpace(queryOrder))
    {
        sql.OrderBy(queryOrder);
    }

    m_db.Query<TData>(sql).ToList().ForEach(data =>
    {
        models.Add(ConvertDataToModel(data));
    });

    m_db.CloseSharedConnection();

    return models;
}
```

Výpis 6: Metoda GetAll() ve třídě BaseModel bez komentářů

3.7.3 Administrace

Dalším úkolem bylo vytvoření administrační části. Zde se měly ukládat jednotlivé kategorie, otázky a odpovědi. Cílem bylo jednoduché a intuitivní rozhraní. Začal jsem tedy postupně od vytváření kategorií, otázek a odpovědí. U kategorie jsem implementoval běžnou validaci modelu a metodu na kontrolu stejného názvu.

V části s otázkami jsem poprvé využil více JavaScriptu s kombinací jQuery a to při změně typu otázky, aby View bylo dynamické a nemusela se celá stránka načítat znova. Zde stojí za zmínku knihovna jQuery [7], která je napsána v JavaScriptu. Usnadňuje vývojáři práci a obsahuje řadu metod, které by si jinak musel psát sám. Nejčastěji ji používám pro procházení HTML dokumentu.

Zákazníkův požadavek na stránku s odpověďmi byl, aby vytváření a editování probíhalo na stejné stránce jako seznam odpovědí. Vytvořil jsem tedy pomocí knihovny jQuery tzv. modal window (vyskakovací okno), kde vkládání probíhalo, nicméně jsem objevil problém s validací na klientské části. Bylo třeba také vytvořit PartialView, které se používá pro rozdělení stránky na více částí, například pro opakující se část kódu. V mém případě bylo vhodné z důvodu použití PartialView v kombinaci s modal window, kde jsem ho načítal pomocí GET požadavku. Problém tedy byl v tom, že PartialView v době vykonávání JavaScriptové validace ještě nebylo zobrazeno a záleželo na uživateli, kdy jej vyvolá. Vyřešil jsem to tak, že jsem pomocí JavaScriptu nefunkční validátory z formuláře pro vytváření a editování odstranil a poté znovu navázal. V PartialView jsem také musel rozeznávat typ odpovědi aby byl vykreslen správný HTML kód. Pro možnost mazání odpovědi jsem v PartialView použil nový formulář se skrytými prvky, které byly pro mazání nutné. Podle požadavků jsem také implementoval tuto funkcionalitu:

- Editace jednotlivé odpovědi na stránce pomocí modal window.
- Kontrola minimálního počtu správných odpovědí a informace uživateli.
- Zobrazení názvu zvolené kategorie a zvolené otázky.

Při vytváření a editování jsem se samozřejmě nemohl spolehnout pouze na validaci v klientské části a tak se vstupy validují i na straně serveru v Controlleru. Při validaci jsem si musel také dávat pozor na speciální znaky, abych se vyhnul XSS útokům. V některých případech bylo ale ukládání a zobrazení takových znaků vyžadováno. Ošetřil jsem tedy vstupní data a speciální znaky jsem nahradil entitami odpovídající danému znaku. Další ošetření bylo proti CSRF útokům. Tato funkcionalita je v ASP.NET implementována, pouze jsem musel zajistit správnou implementaci do formulářů a Controllerů. O samotných útocích XSS a CSRF se můžete více dočíst například na této stránce [8].

Ve výpisu kódu č. 7 můžete vidět vkládání otázky s kódy, validaci modelu pomocí property *ModelState.IsValid* a atribut pro zabezpečení proti CSRF *[ValidateAntiForgeryToken]*.

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Insert(QuestionModel model)
{
    if (ModelState.IsValid)
    {
        int questionId = QuestionModel.Insert(model);

        if (model.QuestionCode.Count > 0)
        {
            for (int i = 0; i < model.QuestionCode.Count; i++)
            {
                model.QuestionCode[i].QuestionId = questionId;
                QuestionCodeModel.Insert(model.QuestionCode[i]);
            }
        }

        return RedirectToAction("Answers", "Answer", new { modelId = questionId });
    }
    else
    {
        ModelState.AddModelError(string.Empty, LanguageResources.ModelError);
    }

    model.QuestionCode = QuestionCodeModel.GetAll(model.Id);
    return View(model);
}
```

Výpis 7: Metoda Insert() ve třídě QuestionController

3.7.4 Generování

V této části jsem měl za úkol zpracovat data do podoby testů. Samozřejmostí bylo také vytvoření Views s Controllery a formuláři. Začal jsem hledat vhodnou knihovnu pro generování PDF v C#. Po delším hledání a diskuzích jsem zvolil knihovnu PDFSharp [9] od empira Software GmbH. Měla přehlednou dokumentaci, ukázky kódu a také byla pod licencí MIT, což byl jeden z hlavních požadavků. S hledáním vhodného řešení pro generování a implemetací jsem strávil nejvíce času. Než jsem se pustil do psaní části s generováním musel jsem si tuto knihovnu nejprve vyzkoušet. Poté jsem přešel k implementaci. Požadavky na stránku pro generování:

- Filtr na zobrazení všech nebo pouze validních kategorií. Validní kategorie byla taková, která měla minimálně 5 otázek a k nim přiřazené odpovědi.
- Výběr jednotlivých kategorií a počtu otázek, které chceme generovat. Počet otázek musel být dělitelý pěti.
- Před samotným generováním zvolit osobu, pro kterou cheme generovat, nebo vytvořit novou.

V této části jsem použil transakce, které slouží pro korektní vkládání více záznamů. Při vytváření záznamu testu se musely korektně provést taky další části, jako například uložení nové osoby. Databáze by byla v nekonzistentním stavu a takový případ nesměl nastat.

Část s generováním byla tedy složitější. Musel jsem nastavit různé možnosti pro korektní vygenerování PDF, zajistit správnost vkládání záznamů a také správné načítání dat pomocí T-SQL funkce.

Na obrázku č. 3 můžete vidět část PDF s testem.

3.7.5 Vyhodnocení a porovnání

V této části jsem měl za úkol vyhodnotit samotný test. Vytvořil jsem tedy novou stránku kde se na základě vyfiltrovaných informací zobrazovaly testy k danému uživateli. Po kliknutí na konkrétní test zde byla stránka s detailem testu. Pokud test nebyl vyhodnocen, byla zde volba *Insert result*. Po zvolení této možnosti se zobrazil seznam otázek a hodnotící osoba označila odpovědi, tak jak byly označené ve vygenerovaném testu. Po kliknutí na tlačítko *Result* se test vyhodnotil, zobrazil správné odpovědi, špatné odpovědi a procentuální výsledek testu. Stejný průběh mělo i editování vyhodnoceného testu. Jediný rozdíl byl v prvním zobrazení, kde byly samozřejmě špatné a správné odpovědi zobrazeny. Po vyhodnocení jsem implementoval generování PDF s výsledky. Generování probíhalo v paměti, jelikož šlo o jednoduchý soubor. Zároveň jsem zde řešil rozdělení tabulky v PDF dokumentu do více stránek. Ve výpisu kódu č. 8 můžete vidět pouze část metody pro vrácení PDF s výsledky. Knihovna PDFSharp umožňuje ukládání do *MemoryStream*, který jsem převedl na pole bajtů. Následně si může uživatel stáhnout požadovaný PDF soubor.

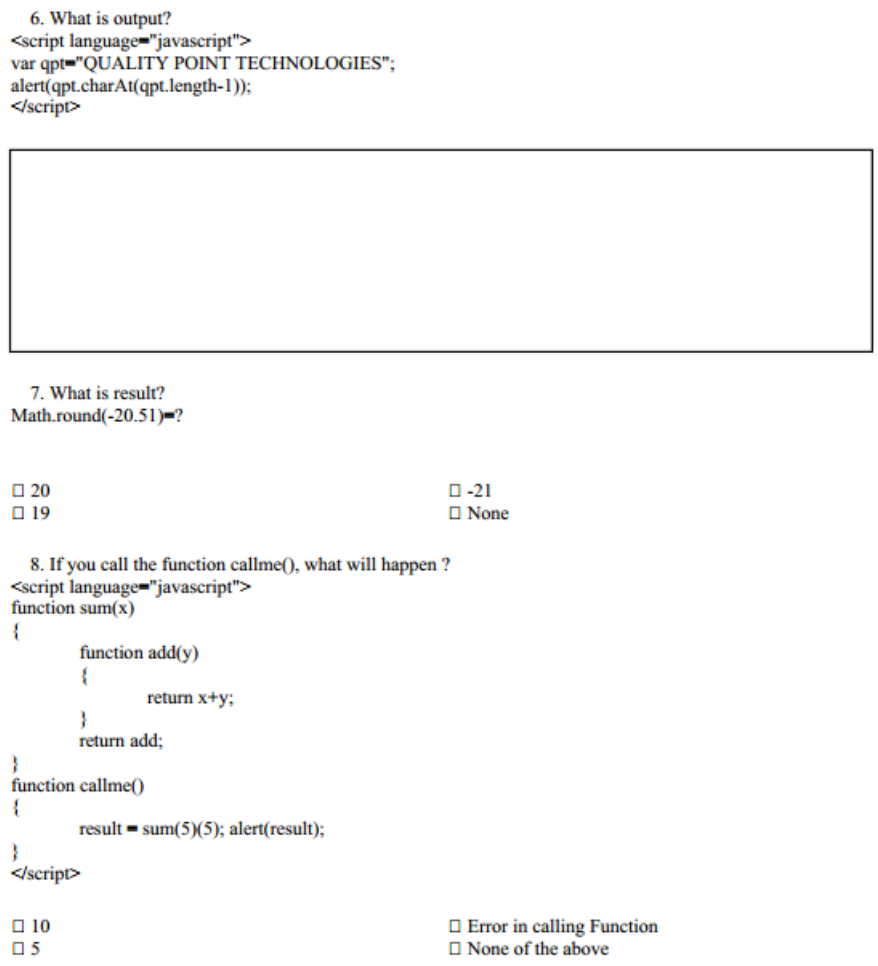
```
using (MemoryStream pdfStream = new MemoryStream())
{
    byte[] fileContents = null;
    pdfRenderer.PdfDocument.Save(pdfStream, true);
    fileContents = pdfStream.ToArray();

    return File(fileContents, "application/pdf", personModel.FirstName + personModel.
        LastName + ".pdf");
}
```

Výpis 8: Část metody pro generování výsledků v PDF

Dalším úkolem bylo najít řešení, jak posílat správná data z View do Controlleru a zpět. Musel jsem znát jedinečný identifikátor odpovědi, správnou odpověď a odpověď testované osoby. Procházením dat v Controlleru a porovnáváním odpovědí jsem vytvořil objekt, který tyto informace obsahoval a na straně klienta jsem napsal JavaScript kód, který dynamicky podle získaného objektu stránku změnil.

Na obrázku č. 4 můžete vidět část View s vyhodnoceným testem.



6. What is output?

```
<script language="javascript">
var qpt="QUALITY POINT TECHNOLOGIES";
alert(qpt.charAt(qpt.length-1));
</script>
```

7. What is result?

Math.round(-20.51)=?

☐ 20 ☐ -21
☐ 19 ☐ None

8. If you call the function callme(), what will happen ?

```
<script language="javascript">
function sum(x)
{
        function add(y)
        {
                return x+y;
        }
        return add;
}
function callme()
{
        result = sum(5)(5); alert(result);
}
</script>
```

☐ 10 ☐ Error in calling Function
☐ 5 ☐ None of the above

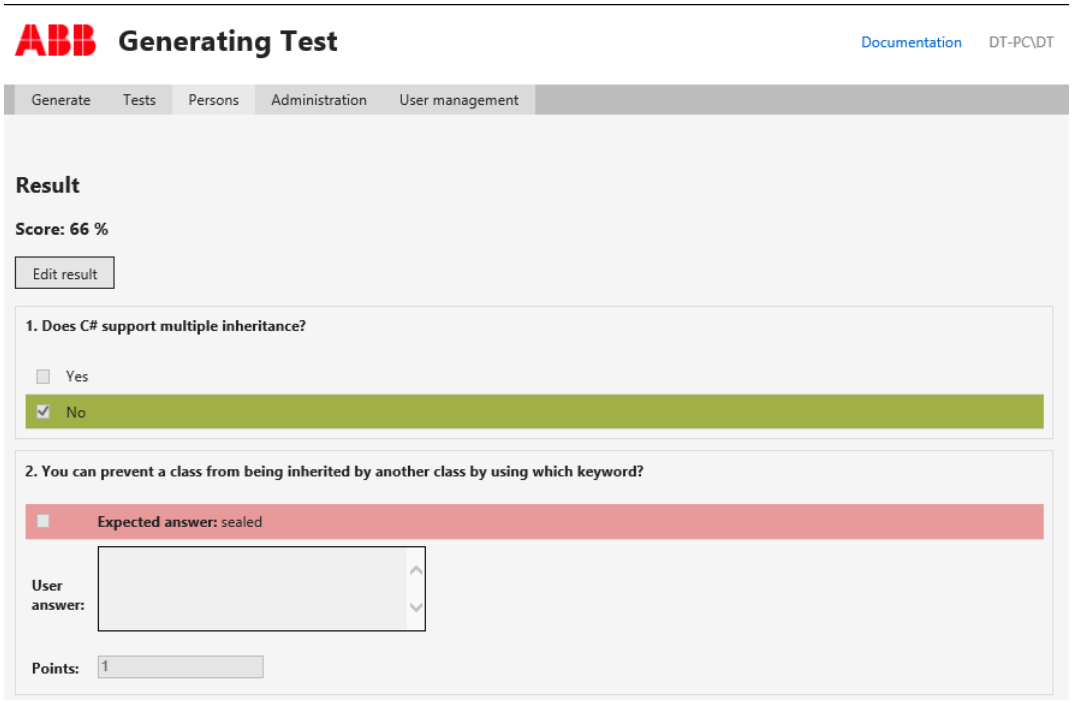
Obrázek 3: Část PDF dokumentu s testem.

3.8 Optimalizace

Při optimalizaci jsem kladl důraz na SQL dotazy a styly CSS.

U SQL dotazů jsem použil SQL Management Studio, zobrazil si plán vykonávání dotazu a snažil se najít co nejlepší způsob jakým dotaz urychlit. V některých situacích si SQL dotaz vyžádal celkově přepsat. Také jsem se snažil vybírat pouze atributy, které jsou opravdu potřeba a vyhnout se dotazům, kde vybírám všechna data aniž bych je kdykoliv použil. Taková optimalizace dokáže ušetřit spoustu času, aplikace bude rychlejší a to i v případě narůstání dat v databázi.

V průběhu projektu jsem se snažil impementovat vzhled aplikace pomocí CSS. Byl jsem však upozorněn, že se nemám spolehnout pouze na testování v jednom prohlížeči. Požadavek byl, aby aplikace byla korektně zobrazena v prohlížeči Internet Explorer ve verzi 9 a novější. Testování jsem však prováděl ze začátku v prohlížeči Google Chrome. Začal jsem tedy s optimalizací CSS, kde jsem kladl důraz na prohlížeč Internet Explorer 9 a



Obrázek 4: Část View s vyhodnoceným testem.

snažil se psát takovým způsobem, aby styly byly univerzální a daly se opakovaně použít. Zachoval jsem také standardy, které mi při psaní byly doporučeny. Výsledný vzhled aplikace z části závisel na mně a byl zhodnocen jako dobrý. Uživatele nijak neobtěžoval a rozložení stránky bylo intuitivní.

Část s optimalizací považuji jako velmi důležitou a také přínosnou.

3.9 Dokumentace

Součástí tohoto projektu byla také uživatelská a programátorská dokumentace, která se měla zpracovávat v průběhu implementace. Tento úkol bylo nutné splnit před nasazením, aby uživatelé byli schopni se systémem pracovat. Programátorská dokumentace se psala nad metody modelů a k částem kódu, u kterých to bylo nutné. Obě dokumentace byly psány v anglickém jazyce.

3.10 Nasazení

Po implementování systému a napsání dokumentace jsem dostal úkol nasadit systém na IIS. IIS je webový server pro Windows, na kterém mohou běžet naše webové aplikace. Měl jsem k dispozici Windows Server 2008 s IIS ve verzi 7 a Microsoft SQL Server. Přesunul jsem tedy můj projekt na tento server a vytvořil zde produkční databázi pomocí nástroje RoundhousE. Ve Visual Studiu jsem musel udělat tzv. *Publish*, který mi vytvořil jeden

soubor s projektem. Před tím jsem ještě musel změnit connection string na produkční databázi. Pravým tlačítkem jsem klikl na projekt s webovým klientem a zvolil možnost *Publish*. Poté jsem vybral metodu *Web deploy package*, zvolil lokaci, kde má být projekt uložený, nastavil název webu a dokončil celou operaci. Po spuštění IIS jsem vygenerovaný balíček importoval. Vytvořila se nová stránka se zadaným názvem, která již byla plně funkční a dostupná pro zaměstnance ABB. Dále bylo nutné otestovat funkčnost celého webu. To znamená, že jsem celý proces od administrace až po vyhodnocení testů musel projít, abych zjistil jestli se někde nevyskytla chyba. Bohužel u některých skriptů byla špatně nastavená adresa pro získávání obsahu a tak se musely tyto soubory přepsat. Dále zde vznikl problém s uložením PDF na server. Bylo mi tedy doporučeno abych PDF generoval v paměti. Po těchto změnách jsem znovu provedl nasazení a otestoval. Vše bylo v pořádku a systém se mohl začít používat.

4 Uplatněné a chybějící znalosti

4.1 Uplatněné znalosti a dovednosti získané v průběhu studia

Na začátku odborné praxe jsem použil znalosti z předmětu Úvod do databázových systémů a to při návrhu databáze. V průběhu to byly znalosti z předmětu Programovací jazyky 2, kde jsem se seznámil s jazykem C#, který jsem používal nejčastěji. Pro psaní složitějších SQL dotazů a správu databáze jsem použil znalosti z předmětu Databázové a informační systémy. U návrhu informačního systému, architektury a vzorů byl pro mě velice přínosný předmět Vývoj informačních systémů, kde jsem se také poprvé dověděl o agilní metodice Scrum.

V jisté míře jsem použil znalosti i z ostatních předmětů, které s výše uvedenými úzce souvisejí. Nejčastěji to byly předměty zaměřené na programovací jazyky. Vysoká škola mi dala velmi dobrý základ pro mou budoucí kariéru a také pro navazující studium.

4.2 Chybějící znalosti a dovednosti v průběhu odborné praxe

V průběhu odborné praxe mi nejvíce chyběla zkušenost práce v týmu a zaměření na zákazníka. Ve škole jsem si většinou projekty musel vymyslet sám nebo byla zveřejněná přesná specifikace. To znamená, že jsem se nemusel nikoho druhého doptávat na požadavky. Na praxi to bylo přesně naopak. Musel jsem se sám doptávat na chybějící informace, které byly nutné a nabízet případně alternativní řešení.

V průběhu studia jsem se také nesetkal s verzovacími systémy pro software, které jsou velmi užitečné. Chyběla mi také větší zkušenost s vytvářením webové aplikace co se týče jejího vzhledu. Ve škole jsem si vybral volitelný předmět Vývoj internetových aplikací, který mi poskytl dobré základy, ale chybělo mi více praktických zkušeností.

Dále jsem rozšiřoval stávající znalosti ze školy a nenarazil jsem na žádnou věc, o které bych nikdy ve škole alespoň neslyšel. Nicméně tyto znalosti byly spíše teoretické a na praxi jsem si je tedy musel osvojit.

5 Závěr

V této práci jsem čtenáře seznámil s firmou ABB, mým zařazením, úkoly, které jsem měl v průběhu praxe vyřešit, a s programy, které jsem použil k vývoji a případně nasazení hotového projektu. Svou odbornou praxi hodnotím pozitivně. Aplikaci se mi podařilo nasadit na produkční server. Používá se k testování nových zaměstnanců a do budoucna jsou plánována rozšíření této aplikace o další funkcionalitu. Naučil jsem se také spoustu nových věcí a přenesl některé teoretické znalosti přímo do praxe. Po ukončení odborné praxe jsem dostal nabídku na pokračování spolupráce s firmou ABB. Je to další možnost, jak získat nové a cenné zkušenosti do budoucna v tak velké firmě.

Na závěr bych chtěl ještě jednou poděkovat firmě ABB za možnost absolvování mé odborné praxe a Vysoké škole báňské za důkladnou přípravu.

Dominik Ther

6 Reference

[1] *Základní údaje o firmě ABB* [online]. ©2015 [cit. 2015-04-21].
Dostupné z: http://new.abb.com/cz/o-nas/zakladni-udaje

[2] *Webové stránky ABB v ČR* [online]. ©2015 [cit. 2015-04-21].
Dostupné z: http://new.abb.com/cz/

[3] *Intro to ASP.NET MVC 4* [online]. 15.8.2012 [cit. 2015-04-21].
Dostupné z: http://www.asp.net/mvc/overview/older-versions/getting-started-with-aspnet-mvc4/intro-to-aspnet-mvc-4

[4] *Project RoundhousE* [online]. 2013 [cit. 2015-04-21].
Dostupné z: https://code.google.com/p/roundhouse/

[5] *Apache License, Version 2.0* [online]. Leden 2004 [cit. 2015-04-21].
Dostupné z: http://www.apache.org/licenses/LICENSE-2.0

[6] *PetaPoco ORM framewwork* [online]. ©2013 [cit. 2015-04-21].
Dostupné z: http://www.toptensoftware.com/petapoco/

[7] *jQuery knihovna* [online]. ©2015 [cit. 2015-04-21].
Dostupné z: http://jquery.com/

[8] *Přehled útoků na webové aplikace* [online]. 10.11.2008 [cit. 2015-04-21].
Dostupné z: http://www.zdrojak.cz/clanky/prehled-utoku-na-webove-aplikace/

[9] *PDFSharp knihovna - empira Software GmbH* [online]. 11.8.2009 11:20 [cit. 2015-04-21].
Dostupné z: http://www.pdfsharp.net/PDFsharpOverview.ashx